

Word-Embeddings Applications

Scalable Data Engineering 2022/23

Julius.Gonsior@tu-dresden.de

Recap

Set of Documents

Then,
he
turned
around

To have
a cat
means
more

First,
he did
not see
it, but



Pre-Processing

then
he
turn
around

have
cat
meaning
more

...

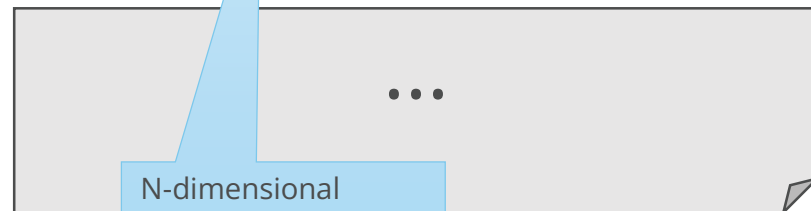
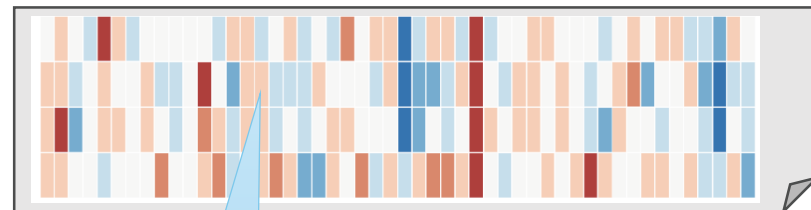
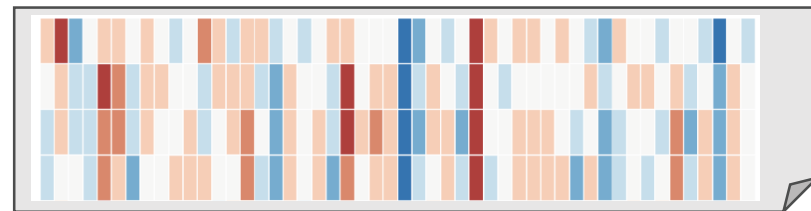
- Tokenization
- Normalization
- Case Folding
- Stop Words
- ...

Multiple
outcomes
possible!

Magic step from
last lecture: using
Neural Network!



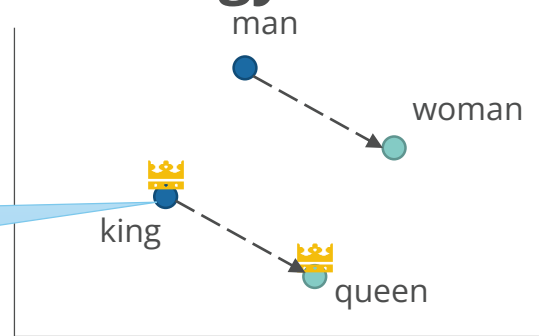
Word Embeddings



N-dimensional
semantic vector
representation per
token

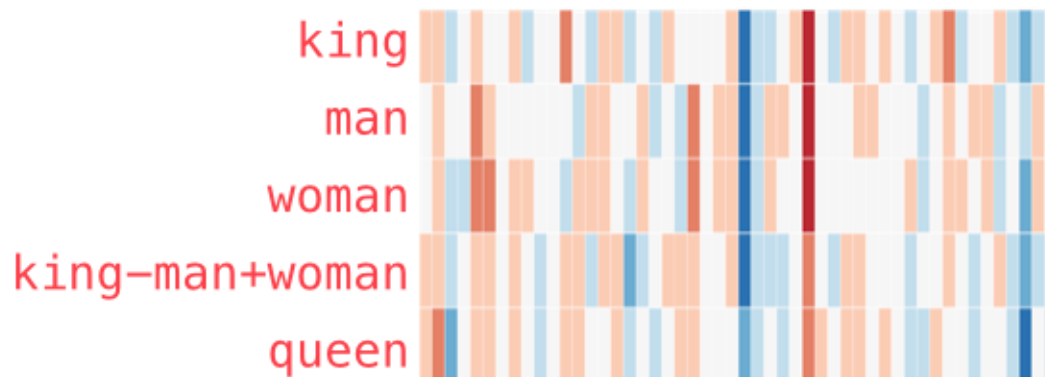
Recap continued: Analogy

Very large N-
dimensional
semantic vector
per token



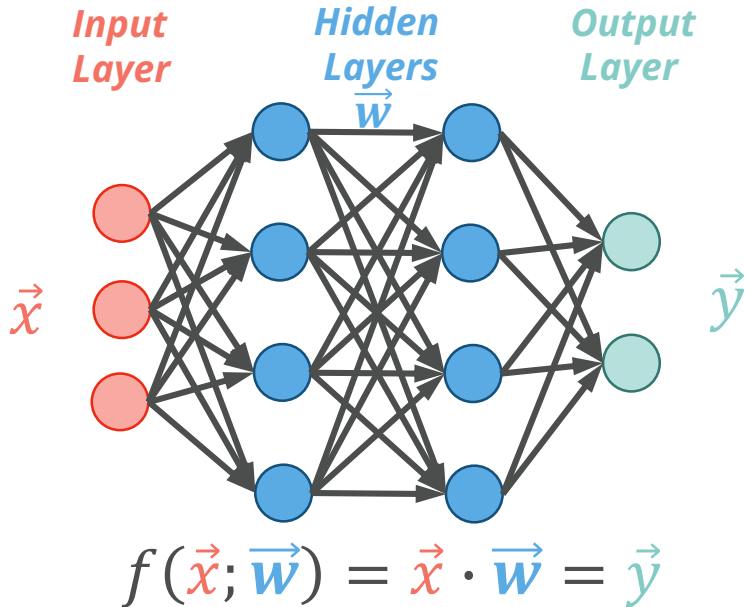
Idea: $\text{man} - \text{woman} = \text{king} - \text{queen}$

$$\text{king} - \text{man} + \text{woman} \approx \text{queen}$$

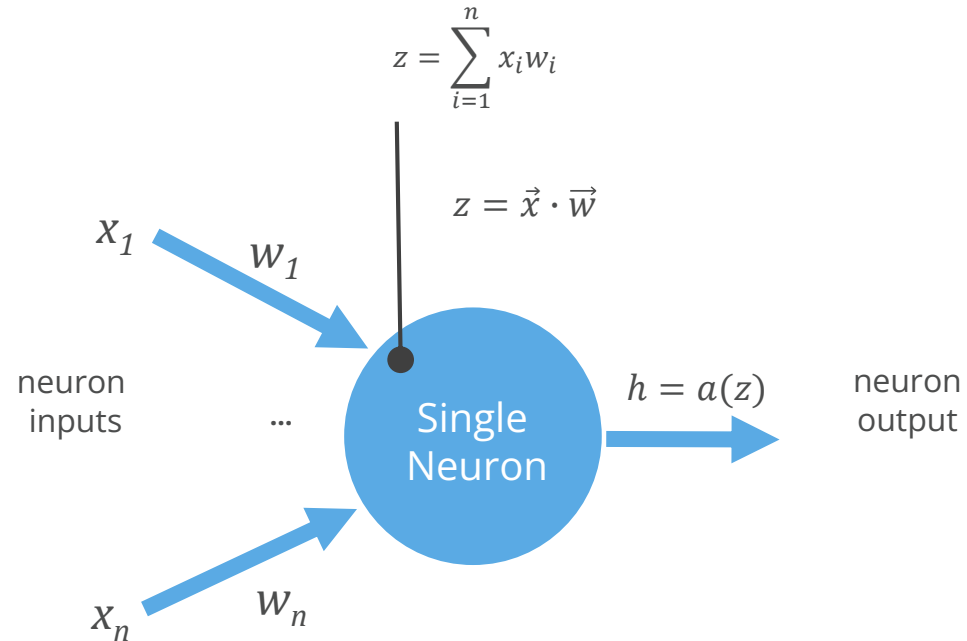


Recap Artificial Neural Networks

Multi-Layer Artificial Neural Network



Single Neuron



Agenda

Relational Databases and Word-Embeddings

Differences

Retrofitting: Synergy

Use case of Word-Embeddings: Text classification

Recurrent Neural
Networks (RNN)

Long-Short
Term Memory (LSTM)

More Use-Cases for LSTMs+Word-Embeddings

4 Point Plan:

- Step 1: Pre-Processing
- Step 2: Word-Embeddings
- **Step 3: ???**
- Step 4: Profit!

Agenda

Relational Databases and Word-Embeddings

Differences

Retrofitting: Synergy

Use case of Word-Embeddings: Text classification

Recurrent Neural
Networks (RNN)

Long-Short
Term Memory (LSTM)

More Use-Cases for LSTMs+Word-Embeddings

Typical Use-Cases of Relational Database

```
SELECT product_desc, price*amount  
FROM products  
ORDER BY 2;
```

```
SELECT ingredients, COUNT(*)  
FROM products  
GROUP BY ingredients  
ORDER BY 2;
```

id	product_desc	ingredients	price	amount
1	chocolate	milk	1.50	27
2	cake	nuts	1.23	12
3	milk	soya	2.07	2
4	burger bun	wheat	8.23	52
...	...	...	...	...
10 ⁷	fish'n'chips	fish	0.89	3

Example relational database with text data

Typical Use-Cases of Relational Database II

```
SELECT products.product_desc, ingredients.sweetnessc  
FROM products  
LEFT JOIN ingredients.ingredients = products.ingredients  
ORDER BY 2;
```

“List all products and the corresponding *sweetness*”

id	product_desc	ingredients	price	amount
1	chocolate	milk	1.50	27
2	cake	nuts	1.23	12
3	milk	soy	2.07	2
4	burger bun	wheat	8.23	52
...	...	...	...	...
10 ⁷	fish'n'chips	fish	0.89	3

Example relational database with text data

id	ingredients	sweetness
1	milk	0.53
2	nuts	0.52
3	soy	0.51
4	wheat	0.42
...	...	...
10 ⁷	fish	0.89

Problems

- Who creates/updates this sweetness table?
- How to measure “sweetness”? Amount of sugar? What about f. e. Stevia?
- Table can becomes very large
- What to do when we have different use-case? Like “List me all ingredients of Mexican cuisine?”...

Better using Word-Embeddings

```
SELECT product_desc
FROM products
ORDER BY similarity(ingredients, 'sweet') DESC;
```

id	product_desc	ingredients	price	amount
1	chocolate	milk	1.50	27
2	cake	nuts	1.23	12
3	milk	soya	2.07	2
4	burger bun	wheat	8.23	52
...	...	...	...	...
10 ⁷	fish'n'chips	fish	0.89	3



product_desc	sweetness
chocolate	0.53
cake	0.52
milk	0.51
burger bun	0.42
...	...

Feature not part of any popular database system
– but it is possible to include word-embeddings out-of-the-box!

Calculation possible without extra “sweetness-ingredient” table -> word embeddings contain already the needed semantic information!

Comparison:

...using Relational Database

- Who **creates/updates** relational tables containing semantic information?
- Tables can become **very large**, expensive to search
- We need **one** table **for each** usage example (e.g. one ingredient-sweetness table, one ingredient-cuisine table, one ingredient-merchant table,...)

...using Word-Embeddings

- Word-Embeddings can be created/updated **automatically** by training on existing text datasets (f.e. Wikipedia articles) for "**free**" -> no human input needed!
- **Weights** of word-embedding neural network $\overrightarrow{w_{embedding}}$ contain all necessary information for word-embedding calculations -> comparably small size!
- Word-Embeddings **contain true semantic knowledge** encoded as numeric vectors -> **universal application** and arithmetic calculations possible!

Agenda

Relational Databases and Word-Embeddings

Differences

Retrofitting: Synergy

Use case of Word-Embeddings: Text classification

Recurrent Neural
Networks (RNN)

Long-Short
Term Memory (LSTM)

More Use-Cases for LSTMs+Word-Embeddings

Retrofitting: Motivation

Word embeddings

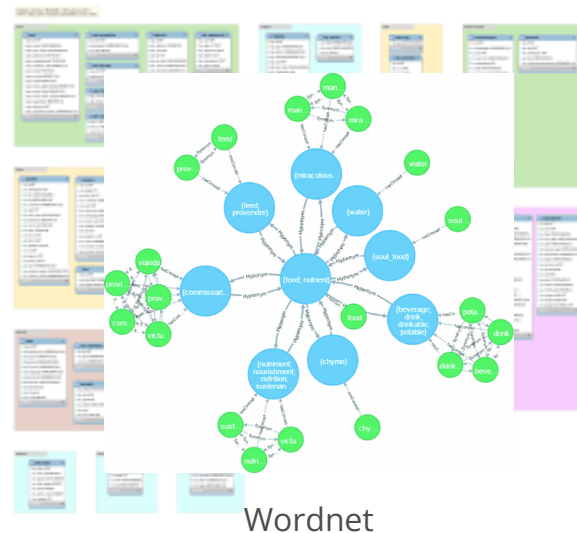
- Powerful and easy to obtain (unsupervised)
- Tend to reflect only similarity (synonymy, connotation)

Can we have the best aspects of both?

- Yes, using Retrofitting (Faruqui et al., 2015)
- Combine knowledge of existing database to further optimize word embeddings
- Iterative update

Structured resources are sparse and hard to obtain...

- ...but they support learning rich, diverse semantic distinctions



Retrofitting: Motivation

Word embeddings

- Powerful and easy to obtain (unsupervised)
- Tend to reflect only similarity (synonymy, connotation)

Can we have the best aspects of both?

- Yes, using Retrofitting (Faruqui et al., 2015)

e.g. Brazil (Country) vs Brazil (Sci-fi Movie)



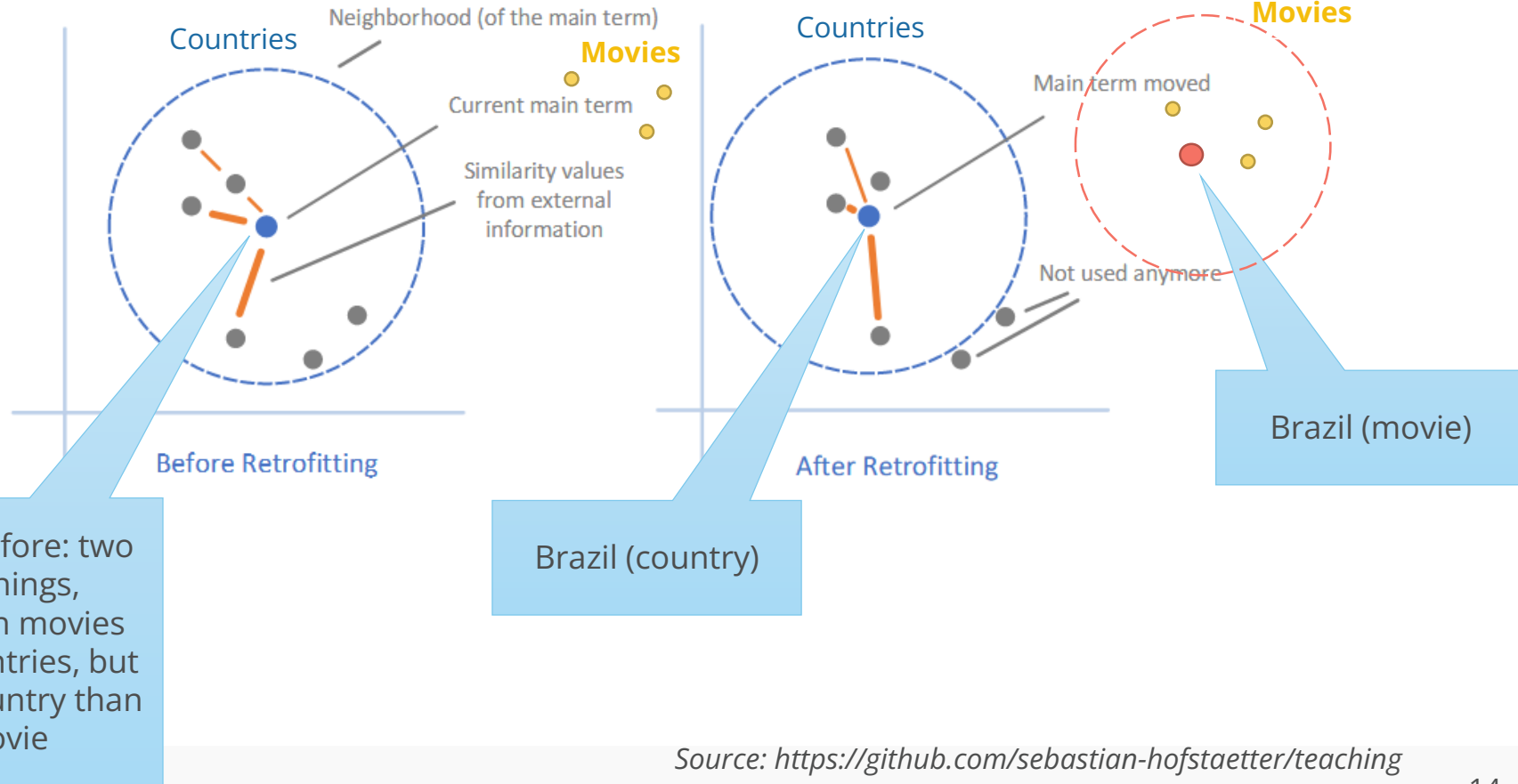
Structured resources are sparse and hard to obtain...

- ...but they support learning rich, diverse semantic distinctions

movie	director	...	released	genre
5th_Element	Luc Besson	...	1997	Sci-Fi
Brazil	Terry Gilliam	...	1995	Dystopia
		...		
Valerian	Luc Besson	...	2017	Space Opera

Columns: entities of the same type/concept

Retrofitting



Source: <https://github.com/sebastian-hofstaetter/teaching>

Agenda

Relational Databases and Word-Embeddings

Differences

Retrofitting: Synergy

Use case of Word-Embeddings: Text classification

Recurrent Neural
Networks (RNN)

Long-Short
Term Memory (LSTM)

More Use-Cases for LSTMs+Word-Embeddings

Motivation

Text Classification example: Language Detection

Other examples:

- Spam detection
- Sentiment analysis (e. g. auto removal of hate comments)
- Urgency Detection (customer feedback)
- Named Entity Recognition
- ...

Then,
he
turned
around

En
svensk
tiger

First,
he did
not see
it, but

Classification

HOW???

English

First,
he did
not see
it, but

Then,
he
turned
around

Swedish

En
svensk
tiger

Classification - How

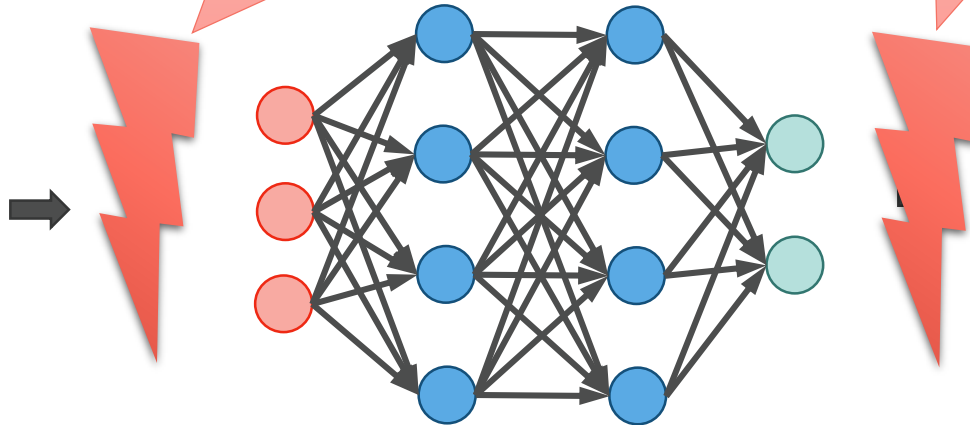
Then,
he
turned
around

En
svensk
tiger

First,
he did
not see
it, but

Problem 1: Neural Networks only understand numerical input

Problem 2: Neural Networks only work with numerical output



Classification Neural Network

Could be replaced by any other supervised ML method as well

English

First,
he did
not see
it, but

Then,
he
turned
around

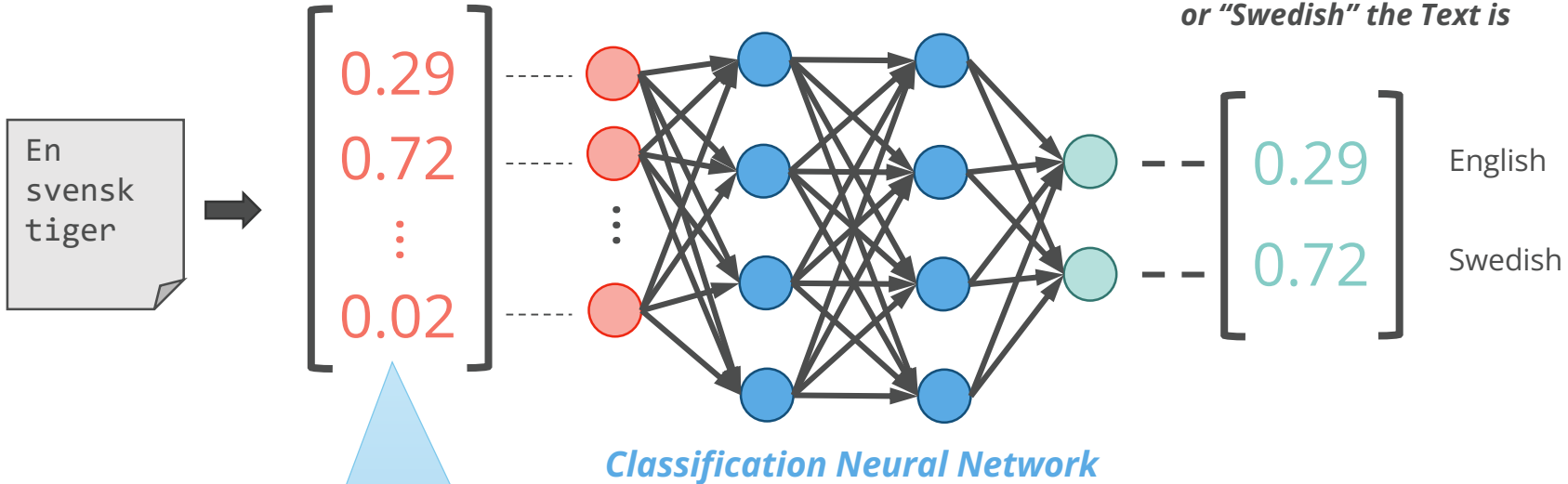
Swedish

En
svensk
tiger

Text Classification using Word Embeddings

Word Embedding -> Input of Neural Network

Output -> Probability of how "English" or "Swedish" the Text is

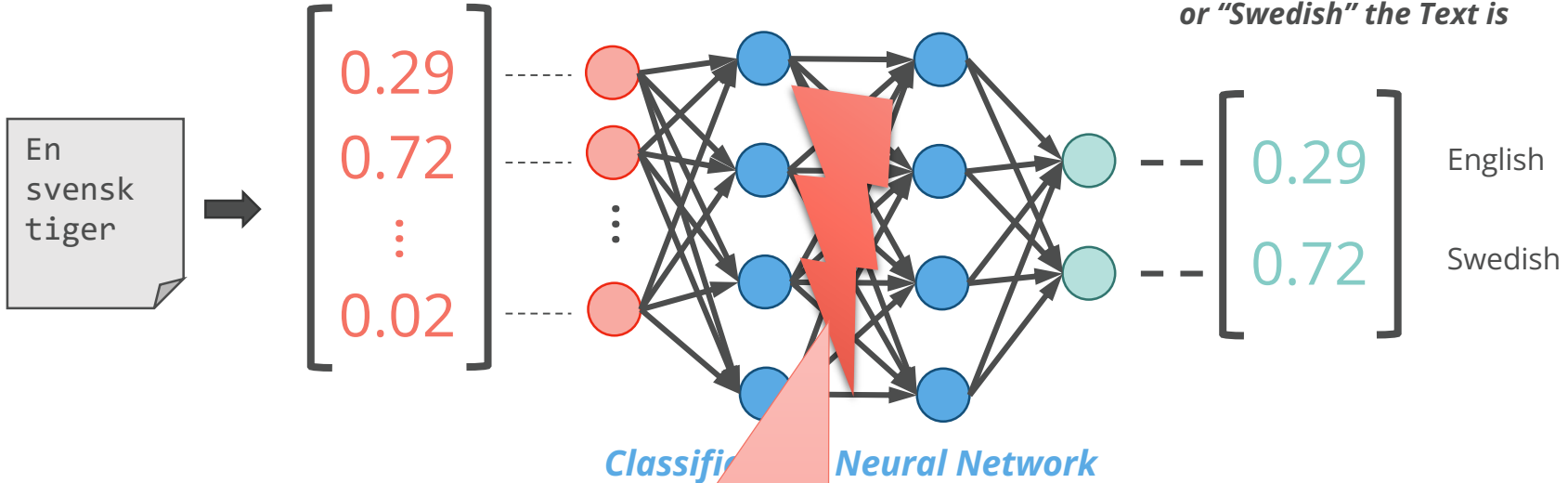


output of the "word-embedding neural network"
AND the input for the
"classification neural network"

Text Classification using Word Embeddings

Word Embedding -> Input of Neural Network

Output -> Probability of how "English"
or "Swedish" the Text is



Problem 3: We have only Word-Embeddings for single words, but documents consist of many words, of on top arbitrary length!

Problem 3: In Detail

Documents

Word Embeddings

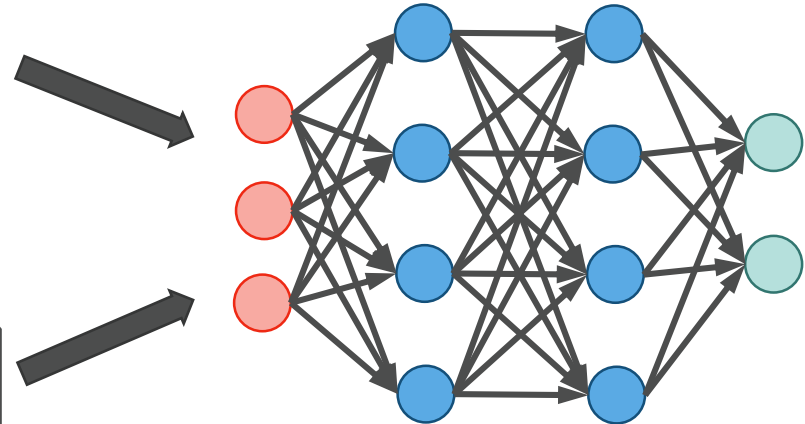
En
svensk
tiger

En	svensk	tiger
0.29	0.31	0.09
0.72	0.57	0.44
⋮	⋮	⋮
0.02	0.92	0.12

Then,
he
turned
around
and

Then	he	turned	around	and
0.25	0.93	0.73	0.15	0.03
0.94	0.75	0.44	0.84	0.16
⋮	⋮	⋮	⋮	⋮
0.35	0.41	0.08	0.16	0.99

How many neurons for **Input Layer**?
 $3 * dim_{word_embeddings}$ or
 $5 * dim_{word_embeddings}$ or something else???



Agenda

Relational Databases and Word-Embeddings

Differences

Retrofitting: Synergy

Use case of Word-Embeddings: Text classification

**Recurrent Neural
Networks (RNN)**

Long-Short
Term Memory (LSTM)

More Use-Cases for LSTMs+Word-Embeddings

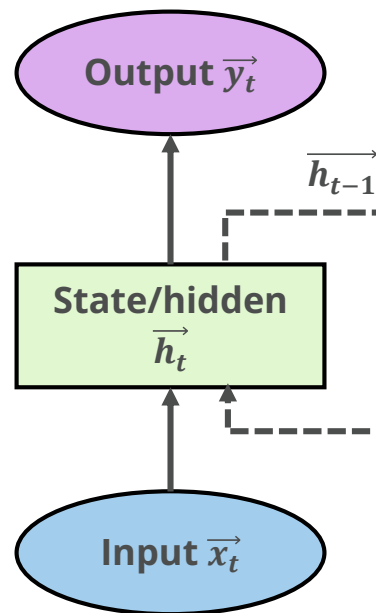
Definition

- RNNs are a special version of Neural Networks which can “remember” inputs -> internal state!
- Optimally suited for sequential input such as time series or text!
- Turing Complete!

“recurrent”

- Neurons not only become $\vec{x}_t$ but also the **previous hidden** state $\vec{h}_{t-1}$ as input

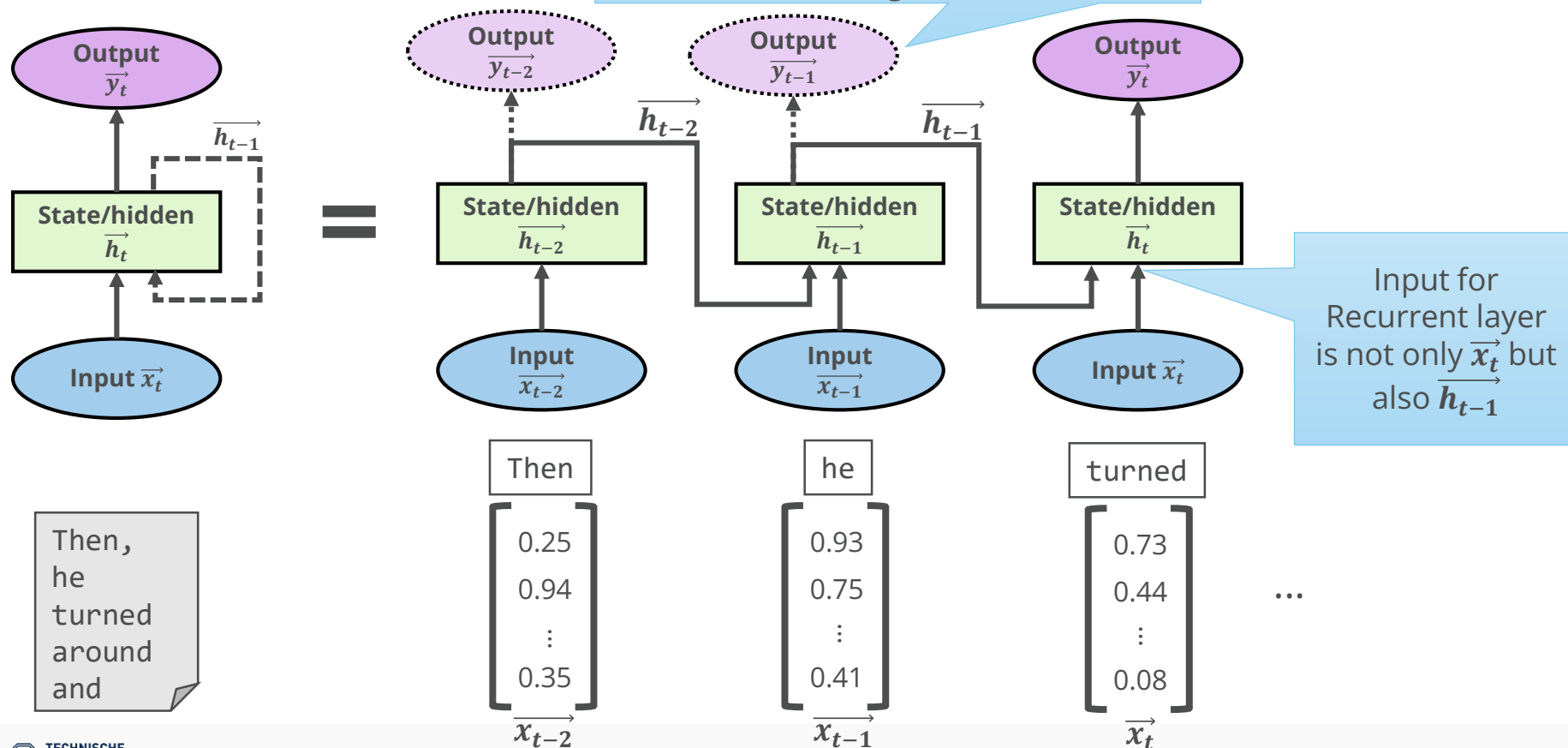
Simple Recurrent Neural Network



Source: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Unrolled/Unfolded RNN

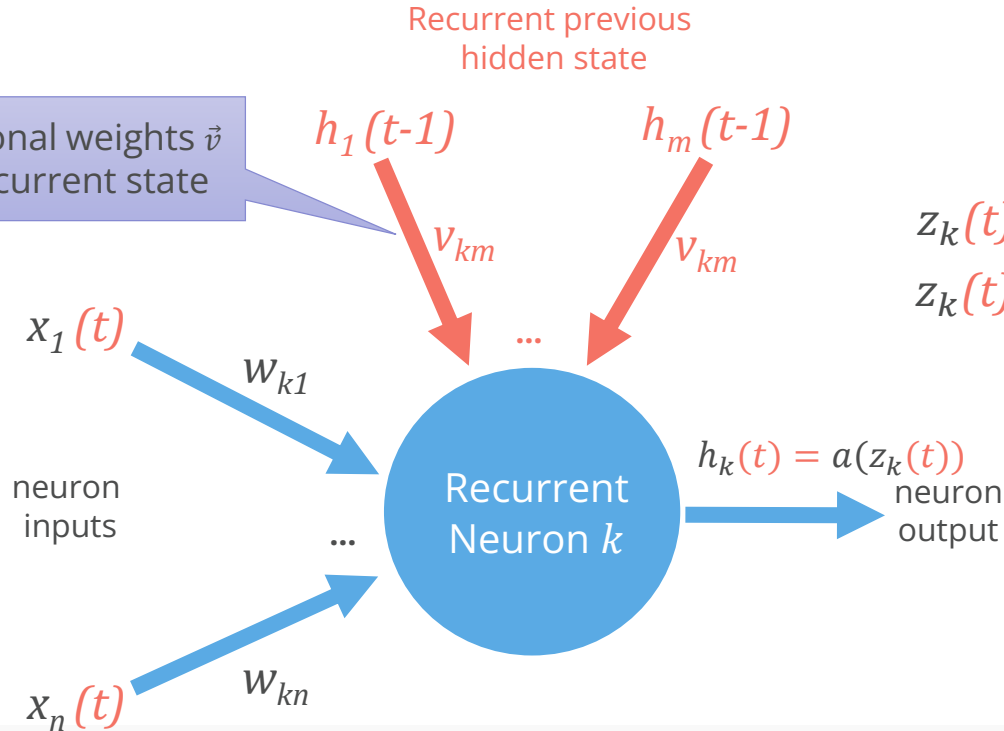
Output at each time step is classification result of seen text so far
-> can change over time!



Recurrent Neuron in detail

Recurrent – new parts

Additional weights $\vec{v}$
for recurrent state



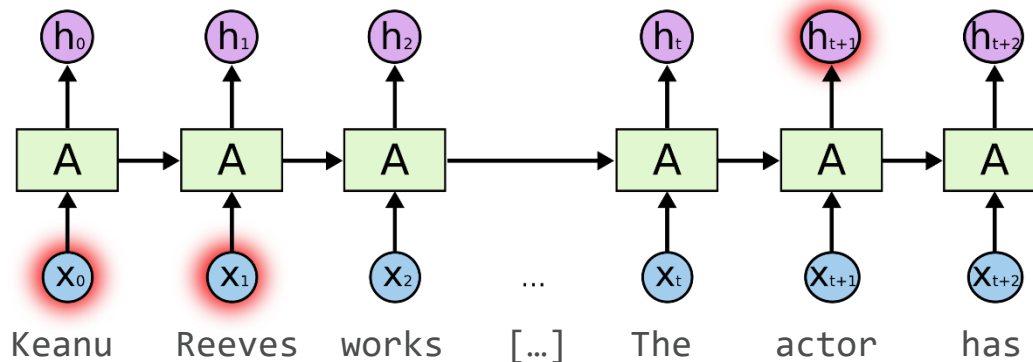
$$z_k(t) = \overrightarrow{x(t)} \cdot \overrightarrow{w} + \overrightarrow{h(t-1)} \cdot \overrightarrow{v}$$

$$z_k(t) = \sum_{i=1}^n x_i(t) w_i + \sum_{j=1}^m h_j(t-1) v_j$$

Problems of RNN

Long-Term Dependencies

- RNNs have only “short-term memory” and often forget beginning of sequence
- Vanishing Gradient problem during backpropagation
 - The more gradients are being multiplied, the higher the chance of one of them being close to zero -> the longer the input sequence the higher the chance that all factors are zero!



Context that “actor” refers to “Keanu Reeves” is often lost after many time steps

Agenda

Relational Databases and Word-Embeddings

Differences

Retrofitting: Synergy

Use case of Word-Embeddings: Text classification

Recurrent Neural
Networks (RNN)

**Long-Short
Term Memory
(LSTM)**

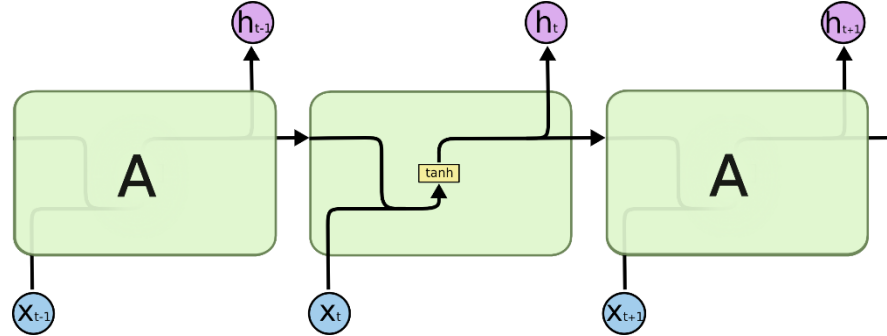
More Use-Cases for LSTMs+Word-Embeddings

Long-Short Term Memory Networks (LSTM)

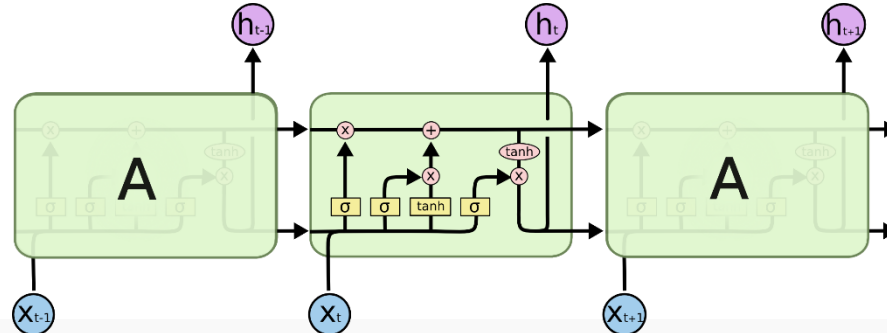
Definition

- LSTM are improved version of RNNs with “long term” memory
- LSTMs consist of 4 different parts:
 - Cell State
 - Input Gate
 - Forget Gate
 - Output Gate

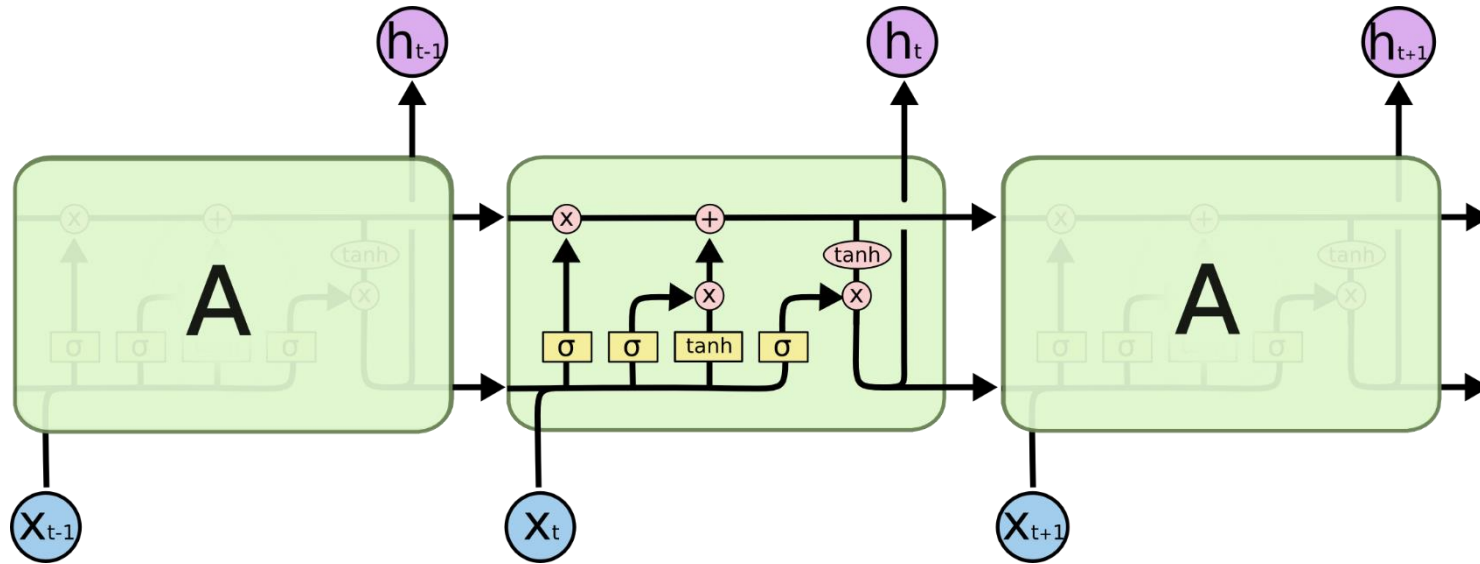
Standard RNN



LSTM



Long-Short Term Memory Networks (LSTM)



- Each line: vector
- Forked line: vector gets duplicated
- Merged line: vector concatenation
- Yellow rectangle: standard neural network layer with each own weights etc.
- h_t is the output/ classification result



Neural Network
Layer



Pointwise
Operation



Vector
Transfer

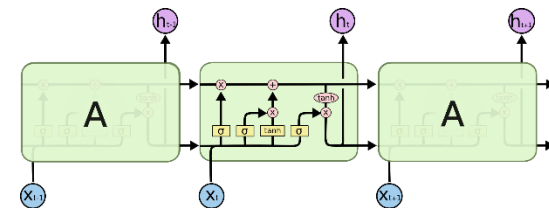
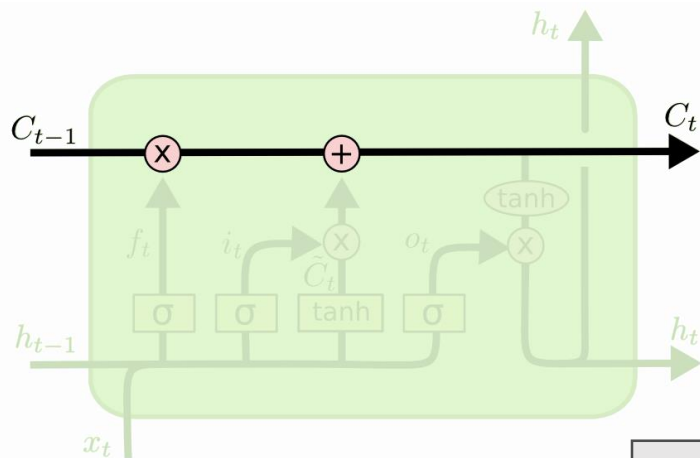


Concatenate



Copy

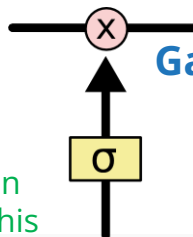
Core Idea behind LSTMs: Cell state



Cell state C_t

- Horizontal line running through top of diagram
- “Memory” of LSTM
- “Gates” control content of cell state
- Information can “pass through” unfolded network
- Ensures that information from early sequential values can pass directly pass through to the current timestamp

En
svensk
tiger

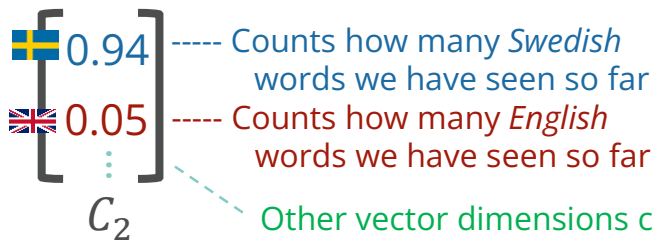


Gates

- Sigmoid activation function
- Output gets added to cell state (pointwise vector operations), can be anything between 0 and 1

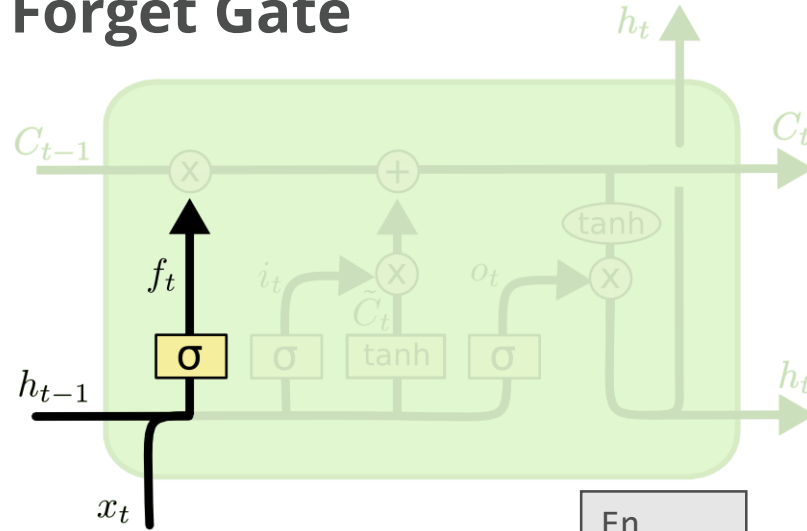
Example t=2

Input after two words: x_1, x_2 : “En”, “svensk”



Other vector dimensions contain information not important for this classification task (maybe the tense of the sentence)

Forget Gate



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

W: weights of this layer
b: bias

Forget Gate Layer

- Decides if information from cell state C_{t-1} is to be discarded
- *Input*: previous hidden state h_{t-1} and x_t
- *Output*: f_t gets pointwise multiplied with cell-state
- 0: "totally forget current cell state"
- 1: "totally keep current cell state"

Example $t = 3$

Input: x_3 : "tiger"

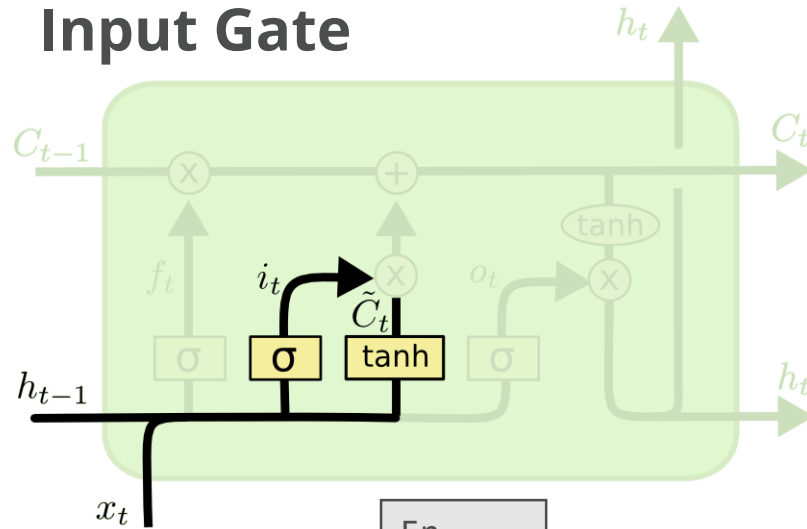
 0.94
 0.05
⋮
 C_2

0.73
0.33
⋮
 f_3

----- Forget a bit what the language was so far
----- Almost completely forget that the text so far was not in English

En
svensk
tiger

Input Gate



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

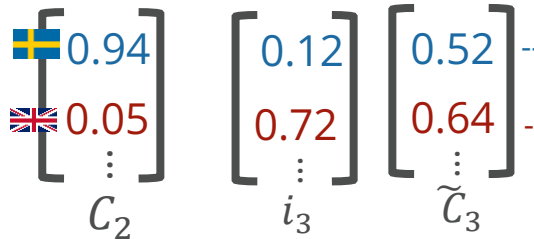
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Input Gate Layer

- Decides which new information to store in cell state C_t
- Sigmoid layer i_t : which values to update?
- tanh layer $\tilde{C}_t$: which new values to add?

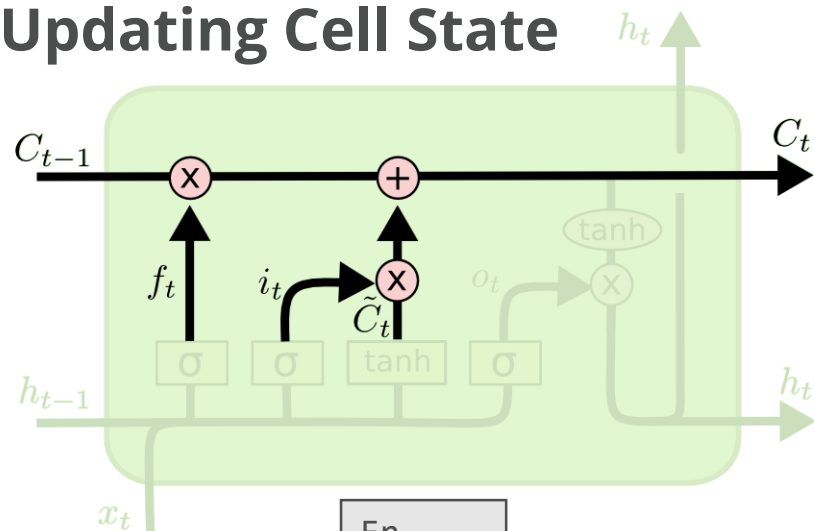
Example t = 3

Input: x_3 : "tiger"



----- tiger does not change our opinion of the "Swedishness" of the input sentence
----- but as it is our first encountered English word we want to update the memory value of "Englishness"

Updating Cell State



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Updating Cell State

- Using output f_t of forget gate, i_t of input gate, and $\tilde{C}_t$ calculate the new cell state C_t
- Note: these are all pointwise vector operations!
- Bascially:
 1. forget old values
 2. add new values, but scaled with i_t
- This prevents “vanishing gradients problem” as we are almost always adding something new to the cell state: $i_t * \tilde{C}_t$

Example $t = 3$

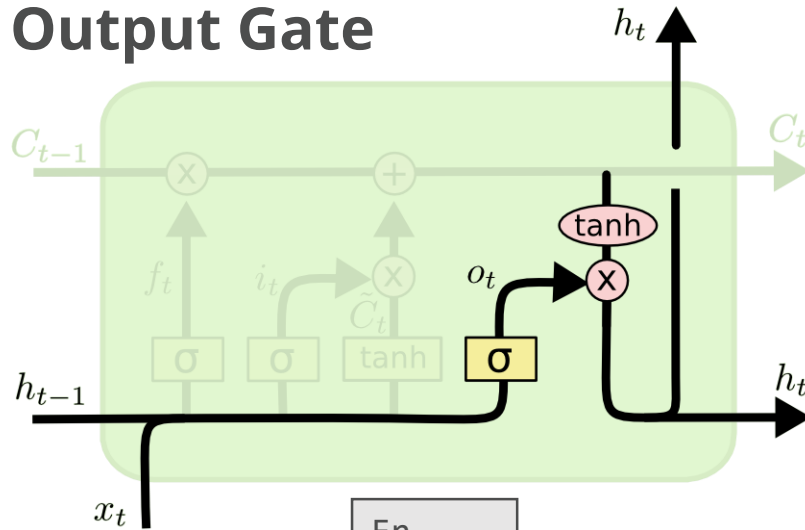
Input: x_3 : “tiger”



$$\begin{bmatrix} 0.75 \\ 0.47 \\ \vdots \end{bmatrix}_{C_3} = \begin{bmatrix} 0.73 \\ 0.33 \\ \vdots \end{bmatrix}_{f_3} * \begin{bmatrix} 0.94 \\ 0.05 \\ \vdots \end{bmatrix}_{C_2} + \begin{bmatrix} 0.12 \\ 0.72 \\ \vdots \end{bmatrix}_{i_3} * \begin{bmatrix} 0.52 \\ 0.64 \\ \vdots \end{bmatrix}_{\tilde{C}_3}$$

C_3 still remembers that the text up to C_2 was mostly Swedish, but still contains the “Englishness” of the newly encountered word

Output Gate



$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh (C_t)$$

Output Gate

- Output h_t is "filtered" version of cell state C_t
- Sigmoid-layer o_t : what parts of cell state are we interested in?

Example $t = 3$

Input: x_3 : "tiger"

 $\begin{bmatrix} 0.75 \\ 0.47 \\ \vdots \end{bmatrix}$
 C_3

$\begin{bmatrix} 0.91 \\ 0.88 \\ \vdots \end{bmatrix}$
 o_3

we are only interested in the first two dimensions of our cell state, therefore both have high values

En
svensk
tiger

$\begin{bmatrix} 0.58 \\ 0.39 \\ \vdots \end{bmatrix}$
 h_3

----- overall we are still sure that the sentence is more Swedish

----- but it could based on the last word also be English

Problem 3: Revisited

Documents

Word Embeddings

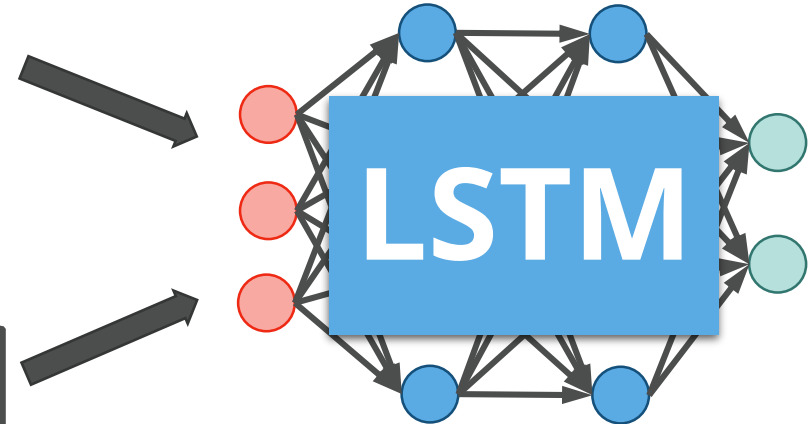
En
svensk
tiger

En	svensk	tiger
0.29	0.31	0.09
0.72	0.57	0.44
⋮	⋮	⋮
0.02	0.92	0.12

Then,
he
turned
around
and

Then	he	turned	around	and
0.25	0.93	0.73	0.15	0.03
0.94	0.75	0.44	0.84	0.16
⋮	⋮	⋮	⋮	⋮
0.35	0.41	0.08	0.16	0.99

How many neurons for **Input Layer**?
 $\dim_{word_embeddings}$ and input each word
vector as sequence



Agenda

Relational Databases and Word-Embeddings

Differences

Retrofitting: Synergy

Use case of Word-Embeddings: Text classification

Recurrent Neural
Networks (RNN)

Long-Short
Term Memory (LSTM)

More Use-Cases for LSTMs+Word-Embeddings

More Application examples (Word Embeddings + LSTM)

Named Entity Recognition (NER)

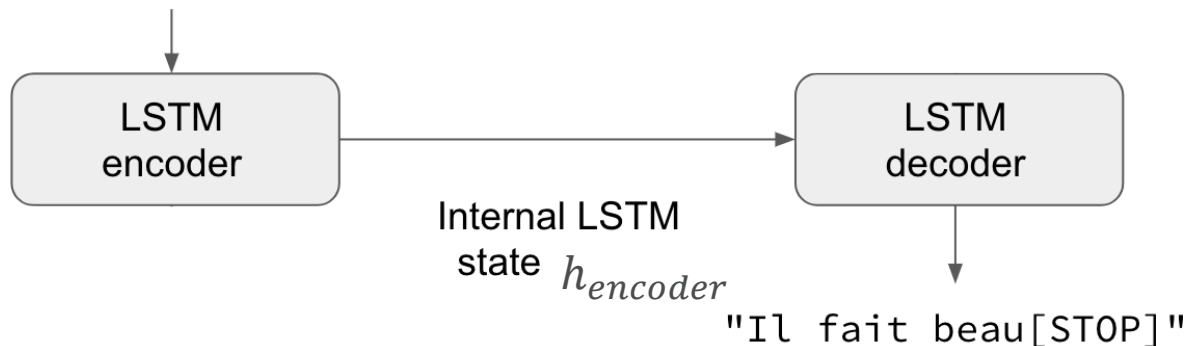
When Sebastian Thrun **PERSON** started working on self-driving cars at Google **ORG** in 2007 **DATE**, few people outside of the company took him seriously.

- Classify (unstructured) text into pre-defined categories
- F1-Scores of over 95% possible: near-human performance!
- *Useful for:*
 - *Auto categorize news/reviews/emails ...*
 - *Extract facts from text*
 - ...

More Application examples (Word Embeddings + LSTM)

Sequence-to-sequence Learning

"The weather is nice"



- Problem: Input and output sequence are of **different** lengths
- Encoder-LSTM:
 - Hidden state $h_{encoder}$ contains dense **fixed-length representation** of input sequence ("context")
- Decoder-LSTM:
 - Constructs **arbitrary** length target sequence from $h_{encoder}$
- *Useful for:*
 - *Translation*
 - *Summarizing*
 - ...

Relational Databases and Word-Embeddings

- Word-Embeddings enable semantic text analyzations without the need for manually curated domain knowledge tables
- Additional to relational databases -> not a replacement!
- Retrofitting: improve general-purpose word-embeddings using domain knowledge from existing relational database

RNNs and LSTMs

- Allow Neural Networks to work on sequential input like text or time series
- LSTM have internal memory to “remember” important information

Topics not covered, but still important for ML based text analysis

Practical Optimizations for Deep Learning Models

- LSTMs can result in very long backpropagation -> optimization needed!
- (mini)-batches, weight sharing, weight initializations, pre-training, dropout,

Convolutional Neural Networks (CNNs)

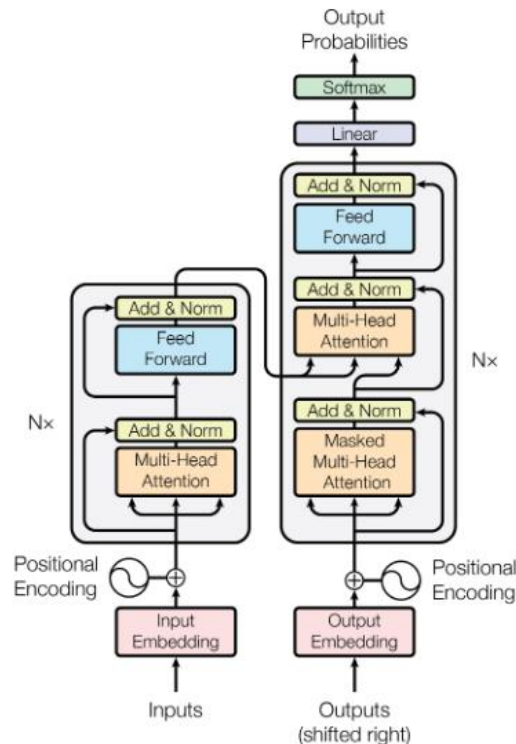
- Mostly used for images -> but also for text!
- parallelizable

Attention Mechanism

- (improved) alternative to LSTMs for sequence-to-sequence learning
- Basic idea: all hidden states from all timestamps are passed to decoder (and get weighted!), not only the last -> works much better for long sentences!

Transformer (BERT etc.)

- Very complex deep neural network architecture built on top of attention idea
- State-of-the-art language model being used today



Deep Learning Criticism

Criticism

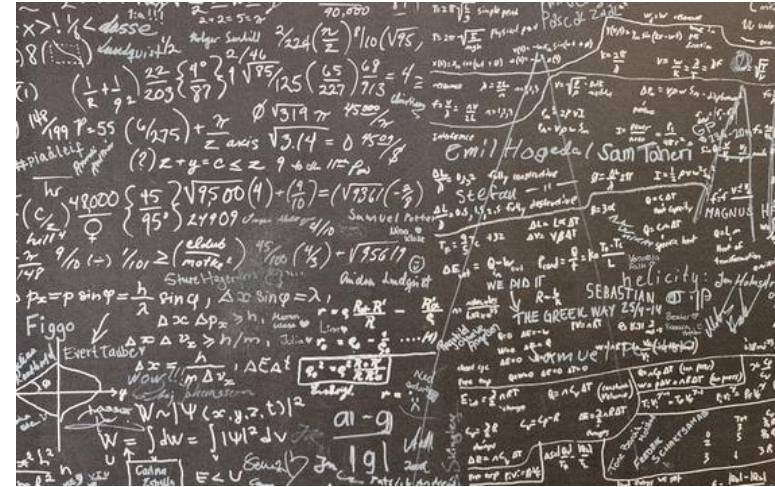
- loooooots of training data (millions) needed -> expensive to get data, expensive to train (multiple GPUs, ...) -> not good for the environment

“Dark Art”/“Black Magic” to train Deep Neural Networks

- Lots of different architecture types, optimizations, data-preprocessing schemes

Lack of extensive mathematical theory

- Still only gradient descent for optimizations
- Theoretically single deep layer is enough for all classification tasks -> in practice not so much!
- Mostly only empirical & heuristic results: “this works” -> but no real clue why!



Neural Networks are NOT the solution to every problem! It is a really hot research topic, but there are also a lot of unfulfilled promises. Often alternative, more simple techniques do the job much better.